

## **AMENDMENTS TO THE CLAIMS**

This listing of claims will replace all prior versions, and listings, of claims in the application:

### **Listing of Claims:**

- 1           1.       (Currently Amended) A method for executing a commit instruction  
2 to facilitate transactional execution on a processor, comprising:  
3           encountering the commit instruction during execution of a program,  
4 wherein the commit instruction marks the end of a block of instructions to be  
5 executed transactionally; and  
6           upon encountering the commit instruction, successfully completing  
7 transactional execution of the block of instructions preceding the commit  
8 instruction, wherein successfully completing the transactional execution involves  
9 atomically committing changes made during the transactional execution by:  
10                 treating store-marked cache lines as locked, thereby causing other  
11                 processes to wait to access the store-marked cache lines;  
12                 committing store buffer entries generated during transactional  
13                 execution to memory, wherein committing each store buffer entry involves  
14                 removing the store-mark from, and thereby unlocking, a corresponding  
15                 store-marked cache line;  
16                 clearing load-marks from cache lines; and  
17                 committing register file changes made during transactional  
18                 execution;  
19           wherein changes made during the transactional execution are not  
20 committed to the architectural state of the processor until the transactional  
21 execution successfully completes.

1           2.       (Previously Presented) The method of claim 1, wherein  
2       successfully completing the transactional execution involves:  
3           resuming normal non-transactional execution.

1           3.       (Cancelled)

1           4.       (Original) The method of claim 1, wherein if an interfering data  
2       access from another process is encountered during the transactional execution and  
3       prior to encountering the commit instruction, the method further comprises:  
4           discarding changes made during the transactional execution; and  
5           attempting to re-execute the block of instructions.

1           5.       (Previously Presented) The method of claim 1, wherein for a  
2       variation of the commit instruction, successfully completing the transactional  
3       execution involves:  
4           commencing transactional execution of the block of instructions following  
5       the commit instruction.

1           6.       (Original) The method of claim 1, wherein potentially interfering  
2       data accesses from other processes are allowed to proceed during the transactional  
3       execution of the block of instructions.

1           7.       (Original) The method of claim 1, wherein the block of instructions  
2       to be executed transactionally comprises a critical section.

1           8.       (Original) The method of claim 1, wherein the commit instruction  
2       is a native machine code instruction of the processor.

1           9.       (Original) The method of claim 1, wherein the commit instruction  
2 is defined in a platform-independent programming language.

1           10.      (Currently Amended) A computer system that supports a commit  
2 instruction to facilitate transactional execution, wherein the commit instruction  
3 marks the end of a block of instructions to be executed transactionally,  
4 comprising:  
5           a processor; and  
6           an execution mechanism within the processor;  
7           wherein upon encountering the commit instruction, the execution  
8 mechanism is configured to successfully complete transactional execution of the  
9 block of instructions preceding the commit instruction, wherein successfully  
10 completing the transactional execution involves atomically committing changes  
11 made during the transactional execution by:  
12                 treating store-marked cache lines as locked, thereby causing other  
13                 processes to wait to access the store-marked cache lines;  
14                 committing store buffer entries generated during transactional  
15                 execution to memory, wherein committing each store buffer entry involves  
16                 removing the store-mark from, and thereby unlocking, a corresponding  
17                 store-marked cache line;  
18                 clearing load-marks from cache lines; and  
19                 committing register file changes made during transactional  
20                 execution;  
21           wherein changes made during the transactional execution are not  
22 committed to the architectural state of the processor until the transactional  
23 execution successfully completes.

1           11.     (Previously Presented) The computer system of claim 10, wherein  
2 while successfully completing transactional execution, the execution mechanism  
3 is configured to:  
4           resume normal non-transactional execution.

1           12.     (Cancelled)

1           13.     (Original) The computer system of claim 10, wherein if an  
2 interfering data access from another process is encountered during the  
3 transactional execution and prior to encountering the commit instruction, the  
4 execution mechanism is configured to:  
5           discard changes made during the transactional execution; and to  
6           attempt to re-execute the block of instructions.

1           14.     (Previously Presented) The computer system of claim 10, wherein  
2 if a variation of the commit instruction is encountered, the execution mechanism  
3 is configured to:  
4           commence transactional execution of the block of instructions following  
5 the commit instruction.

1           15.     (Original) The computer system of claim 10, wherein the computer  
2 system is configured to allow potentially interfering data accesses from other  
3 processes to proceed during the transactional execution of the block of  
4 instructions.

1           16.     (Original) The computer system of claim 10, wherein the block of  
2 instructions to be executed transactionally comprises a critical section.

1           17.     (Original) The computer system of claim 10, wherein the commit  
2     instruction is a native machine code instruction of the processor.

1           18.     (Original) The computer system of claim 10, wherein the commit  
2     instruction is defined in a platform-independent programming language.

1           19.     (Currently Amended) A computer-readable storage medium storing  
2     instructions that when executed by a computer cause the computer to perform a  
3     method for executing a commit instruction to facilitate transactional execution,  
4     comprising:  
5             encountering the commit instruction during execution of a program,  
6     wherein the commit instruction marks the end of a block of instructions to be  
7     executed transactionally; and  
8             upon encountering the commit instruction, successfully completing  
9     transactional execution of the block of instructions preceding the commit  
10    instruction, wherein successfully completing the transactional execution involves  
11    atomically committing changes made during the transactional execution by:  
12             treating store-marked cache lines as locked, thereby causing other  
13             processes to wait to access the store-marked cache lines;  
14             committing store buffer entries generated during transactional  
15             execution to memory, wherein committing each store buffer entry involves  
16             removing the store-mark from, and thereby unlocking, a corresponding  
17             store-marked cache line;  
18             clearing load-marks from cache lines; and  
19             committing register file changes made during transactional  
20             execution;

21            wherein changes made during the transactional execution are not  
22 committed to the architectural state of the processor until the transactional  
23 execution successfully completes.

1            20.    (Previously Presented) The computer-readable storage medium of  
2 claim 19, wherein successfully completing transactional execution involves:  
3            resuming normal non-transactional execution.